Sparse multivariate polynomial interpolation in the basis of Schubert polynomials

Priyanka Mukhopadhyay [*] Youming Qiao [†]

September 30, 2016

Abstract

Schubert polynomials were discovered by A. Lascoux and M. Schützenberger in the study of cohomology rings of flag manifolds in 1980's. These polynomials generalize Schur polynomials, and form a linear basis of multivariate polynomials. In 2003, Lenart and Sottile introduced skew Schubert polynomials, which generalize skew Schur polynomials, and expand in the Schubert basis with the generalized Littlewood-Richardson coefficients.

In this paper we initiate the study of these two families of polynomials from the perspective of computational complexity theory. We first observe that skew Schubert polynomials, and therefore Schubert polynomials, are in #P (when evaluating on non-negative integral inputs) and VNP.

Our main result is a deterministic algorithm that computes the expansion of a polynomial f of degree d in $\mathbb{Z}[x_1, \ldots, x_n]$ in the basis of Schubert polynomials, assuming an oracle computing Schubert polynomials. This algorithm runs in time polynomial in n, d, and the bit size of the expansion. This generalizes, and derandomizes, the sparse interpolation algorithm of symmetric polynomials in the Schur basis by Barvinok and Fomin (Advances in Applied Mathematics, 18(3):271–285). In fact, our interpolation algorithm is general enough to accommodate any linear basis satisfying certain natural properties.

Applications of the above results include a new algorithm that computes the generalized Littlewood-Richardson coefficients.

1 Introduction

Polynomial interpolation problem. The classical polynomial interpolation problem starts with a set of data points, (a_1, b_1), ..., (a_{n+1}, b_{n+1}), where $a_i, b_i \in \mathbb{Q}$, $a_i \neq a_j$ for $i \neq j$, and asks for a univariate polynomial $f \in \mathbb{Q}[x]$ s.t. $f(a_i) = b_i$ for $i = 1, \ldots, n+1$. While such a polynomial of degree $\leq n$ exists and is unique, depending on different choices of linear bases, several formulas, under the name of Newton, Lagrange, and Vandermonde, have become classical.

In theoretical computer science (TCS), the following problem also bears the name interpolation of polynomials, studied from 1970's: given a black-box access to a multivariate polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ ($\mathbb{F}$ a field), compute f by writing out its sum-of-monomials expression. Several

[*] Center for Quantum Technologies, National University of Singapore (mukhopadhyay.priyanka@gmail.com).

[†] Centre for Quantum Computation and Intelligent Systems, University of Technology, Sydney, Australia (jimmyqiao86@gmail.com).

algorithms have been proposed to solve this problem [Zip79, BOT88, KY88, Zip90, KS01]. A natural generalization is to consider expressing f using the more powerful arithmetic circuits. In this more general setting, the problem is called the reconstruction problem for arithmetic circuits [SY10, Chap. 5], and the sum-of-monomials expression of f is viewed as a subclass of arithmetic circuits, namely depth-2 $\Sigma\Pi$ circuits. Reconstruction problems for various models gained quite momentum recently [BC98, Shp09, SV08, KS09, AMS10, GKL11, Kay12, GKL12, GKQ14].

As mentioned, for interpolation of univariate polynomials, different formulas depend on different choices of linear bases. On the other hand, for interpolation (or more precisely, reconstruction) of multivariate polynomials in the TCS setting, the algorithms depend on the computation models crucially. In the latter context, to our best knowledge, only the linear basis of monomials (viewed as depth-2 $\Sigma\Pi$ circuits) has been considered.

Schubert polynomials. In this paper, we consider the interpolation of multivariate polynomials in the TCS setting, but in another linear basis of multivariate polynomials, namely the Schubert polynomials. This provides another natural direction for generalizing the multivariate polynomial interpolation problem. Furthermore, as will be explained below, such an interpolation algorithm can be used to compute certain quantities in geometry that are of great interest, yet not well-understood.

Schubert polynomials were discovered by Lascoux and Schützenberger [LS82] in the study of cohomology rings of flag manifolds in 1980's. See Definition 8 or Definition 12 for the definition[1], and [Mac91, Man01] for detailed results. For now we only point out that (1) Schubert polynomials in $\mathbb{Z}[x_1, \ldots, x_n]$ are indexed by $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{N}^n$, denoted by $Y_{\mathbf{v}}$;[2] (2) $Y_{\mathbf{v}}$ is homogeneous of degree $\sum_{i=1}^n v_i$.[3]

Schubert polynomials have many distinguished properties. They form a linear basis of multivariate polynomials, and yield a generalization of Newton interpolation formula to the multivariate case [Las03, Sec. 9.6]. Also, Schur polynomials are special Schubert polynomials (13 (1)). A Schubert polynomial can contain exponentially many monomials: the complete homogeneous symmetric polynomials are special Schubert polynomials (13 (2)). It is not clear to us whether Schubert polynomials have polynomial-size arithmetic circuits. Because of these reasons, interpolation in the Schubert basis could not be covered by the aforementioned results for the reconstruction problems, unless the arithmetic circuit complexity of Schubert polynomials is understood better: at present, we are only able to put Schubert polynomials in VNP.

While Schubert polynomials are mostly studied due to their deep geometric meanings (see e.g. [KM05]), they do have certain algorithmic aspects that have been studied shortly after their introduction in 1982. Indeed, an early paper on Schubert polynomials by Lascoux and Schützenberger was concerned about using them to compute the Littlewood-Richardson coefficients [LS85]. That procedure has been implemented in the program system `Symmetrica` [KKL92], which includes a set of routines to work with Schubert polynomials. On the other hand, the complexity-theoretic study of the algorithmic aspects of Schubert polynomials seems lacking, and we hope that this paper

[1] Definition 12 is one of the classical definitions of Schubert polynomials, while Definition 8 defines Schubert polynomials in the context of skew Schubert polynomials.

[2] In the literature, it is more common that Schubert polynomials indexed by permutations instead of $\mathbb{N}^n$. These two index sets are equivalent, through the correspondence between permutations and $\mathbb{N}^n$ as described in Section 2. We adopt $\mathbb{N}^n$ in the introduction because they are easier to work with when dealing with a fixed number of variables.

[3] Algorithms in this work run in time polynomial in the degree of the polynomial. By (2) this is equivalent to that the indices of Schubert polynomials are given in unary.

serves as a modest step towards this direction.

Our results. Our main result is about deterministic interpolation of sparse polynomials with integer coefficients in the Schubert basis, modulo an oracle that computes Schubert polynomials. The complexity is measured by the bit size of the representation.

Theorem 1. *Suppose we are given (1) black-box access to some polynomial* $f \in \mathbb{Z}[x_1, \ldots, x_n]$, $f = \sum_{\mathbf{v}\in\Gamma} a_{\mathbf{v}} Y_{\mathbf{v}}$, $a_{\mathbf{v}} \neq 0 \in \mathbb{Z}$ *with the promise that* $\deg(f) \leq d$ *and* $|\Gamma| \leq m$*; (2) an oracle that computes the evaluation of Schubert polynomials on nonnegative integral points. Then there exists a deterministic algorithm, that outputs the expansion of* f *in the basis of Schubert polynomials. The algorithm runs in time polynomial in* n, d, m, *and* $\log(\sum_{\mathbf{v}\in\Gamma} |a_{\mathbf{v}}|)$.

In fact, Theorem 1 relies on the algorithm in Theorem 9 which applies to a more general setting: it only requires the linear basis where the leading monomials are "easy to isolate." See *Our techniques* for a more detailed discussion.

Theorem 1 generalizes and derandomizes a result by Barvinok and Fomin, who in [BF97] present a randomized algorithm that interpolates sparse symmetric polynomials in the Schur basis. As mentioned, Schur polynomials are special Schubert polynomials, and the Jacobi-Trudi formulas for Schur polynomials yield efficient algorithms to compute them. So to recover Barvinok and Fomin's result, apply our interpolation algorithm to symmetric polynomials, and replace the #P oracle computing Schubert polynomials by the efficient algorithm computing Schur polynomials. Likewise, for those Schubert polynomials with efficient evaluation procedures[4], we can get rid of the #P oracle to obtain a polynomial-time interpolation algorithm.

Our second result concerns the evaluation of Schubert polynomials. In fact, we shall work with a generalization of Schubert polynomials, namely *skew Schubert polynomials* as defined by Lenart and Sottile [LS03]. We will describe the definition in Section 2. For now, we only remark that skew Schubert polynomials generalize Schubert polynomials in a way analogous to how skew Schur polynomials generalize Schur polynomials. A skew Schubert polynomial, denoted by $Y_{\mathbf{w}/\mathbf{v}}$, is indexed by $\mathbf{v} \leq \mathbf{w} \in \mathbb{N}^m$ where $\leq$ denotes the Bruhat order[5]. Schubert polynomials can be defined by setting $\mathbf{w}$ to correspond to the permutation maximal in the Bruhat order.

Theorem 2. *Given* $\mathbf{v}, \mathbf{w} \in \mathbb{N}^m$ *in unary, and* $\mathbf{a} \in \mathbb{N}^n$ *in binary, computing* $Y_{\mathbf{w}/\mathbf{v}}(\mathbf{a})$ *is in #P.*

Corollary 3. *Given* $\mathbf{v} \in \mathbb{N}^n$ *in unary, and* $\mathbf{a} \in \mathbb{N}^n$ *in binary, computing* $Y_{\mathbf{v}}(\mathbf{a})$ *is in #P.*

Note that in Theorem 2 we have $\mathbf{v}, \mathbf{w} \in \mathbb{N}^m$, while in Corollary 3 we have $\mathbf{v} \in \mathbb{N}^n$. This is because, while for Schubert polynomials we know $Y_{\mathbf{v}}$ for $\mathbf{v} \in \mathbb{N}^n$ depends on n variables, such a relation is not clear for skew Schubert polynomials.

Finally, we also study these polynomials in the framework of algebraic complexity.

Theorem 4. *Skew Schubert polynomials, and therefore Schubert polynomials, are in* VNP.

[4] For example, there are determinantal formulas [Man01, Sec. 2.6] for Schubert polynomials indexed by 2143-avoiding permutations ($\nexists i < j < k < \ell$ s.t. $\sigma(j) < \sigma(i) < \sigma(\ell) < \sigma(k)$), and 321-avoiding permutations ($\nexists i < j < k$ s.t. $\sigma(k) < \sigma(j) < \sigma(i)$). 2143-avoiding permutations are also known as vexillary permutations and form a generalization of Grassmannian permutations.

[5] Bruhat order on codes is inherited from the Bruhat order on permutations through the correspondence between codes and permutations as in Section 2.

Applications of our algorithms. A long-standing open problem about Schubert polynomials is to give a combinatorially positive rule, or in other words, a #P algorithm for the *generalized Littlewood-Richardson (LR) coefficients*, defined as the coefficients of the expansion of products of two Schubert polynomials in the Schubert basis. They are also the coefficients of the expansion of skew Schubert polynomials in the Schubert basis [LS03]. These numbers are of great interest in algebraic geometry, since they are the intersection numbers of Schubert varieties in the flag manifold. See [ABS14, MPP14] for recent developments on this.

The original LR coefficients are a special case when replacing Schubert with Schur in the above definition. It is known that the original LR coefficients are #P-complete, by the celebrated LR rule, and a result of [Nar06]. Therefore the generalized LR coefficients are also #P-hard to compute, while as mentioned, putting this problem in #P is considered to be very difficult – in fact, we were not aware of any non-trivial complexity-theoretic upper bound. On the other hand, by interpolating skew Schubert polynomials in the Schubert basis, we have the following.

Corollary 5. *Given* $\mathbf{w}, \mathbf{v} \in \mathbb{N}^m$*, let* $\Gamma = \{\mathbf{u} \in \mathbb{N}^m \mid a_{\mathbf{w}}^{\mathbf{v},\mathbf{u}} \neq 0\}$*. Then there exists a deterministic algorithm that, given access to an #P oracle, computes* $(a_{\mathbf{w}}^{\mathbf{v},\mathbf{u}} | \mathbf{u} \in \mathbb{N}^m)$ *in time polynomial in* $|\mathbf{w}|, |\Gamma|$*, and* $\log(\sum_{\mathbf{u}\in\Gamma} a_{\mathbf{w}}^{\mathbf{v},\mathbf{u}})$*.*

The algorithm in Corollary 5 has the benefit of running in time polynomial in the *bit size* of $a_{\mathbf{w}}^{\mathbf{v},\mathbf{u}}$. Therefore, when $|\Gamma|$ is small compared to $\sum_{\mathbf{w}} a_{\mathbf{w}}^{\mathbf{v},\mathbf{u}}$, our algorithm is expected to lead to a notable saving, compared to those algorithms that are solely based on positive rules, e.g. [Kog01]. ([Kog01] furthermore only deals with the case of Schubert times Schur.) Of course, in practice we need to take into account the time for evaluating Schubert polynomials.

In addition, we note that Barvinok and Fomin's original motivation is to compute e.g. the Littlewood-Richardson coefficients, Kostka numbers, and the irreducible characters of the symmetric group. See [BF97, Sec. 1] for the definitions and importance of these numbers. Since our algorithm can recover theirs (without referring to a #P oracle), it can be used to compute these quantities as well. Note that our algorithm is moreover deterministic.

Our original motivation of this work was to better understand this approach of Barvinok and Fomin to compute the LR coefficients. This topic recently receives attention in complexity theory [Nar06, MNS12, BI13], due to its connection to the geometric complexity theory (GCT) [Mul11, MNS12]. Though this direction of generalization does not apply to GCT directly, we believe it helps in a better understanding (e.g. a derandomization) of this approach of computing the LR coefficients.

Our techniques. We achieve Theorem 1 by first formalizing some natural properties of a linear basis of (subrings of) multivariate polynomials (Section 4.1). These are helpful for the interpolation purpose. If a basis satisfies these properties, we call this basis *interpolation-friendly*, or I-friendly for short. Then we present a deterministic interpolation algorithm for I-friendly bases (Theorem 9). We then prove that the Schubert basis is interpolation-friendly[6] (Section 5).

Technically, for the interpolation algorithm, we combine the structure of the Barvinok-Fomin algorithm with several ingredients from the elegant deterministic interpolation algorithm for sparse

[6] While the proofs for these properties of Schubert polynomials are easy, and should be known by experts, we include complete proofs as we could not find complete proofs or explicit statements. Most properties of Schubert polynomials, e.g. the #P result, Lemma 14, Proposition 17, can also be obtained by a combinatorial tool called RC graphs [BB93]. We do not attempt to get optimal results (e.g. in Corollary 16 and Proposition 17), but are content with bounds that are good enough for our purpose.

polynomials in the monomial basis by Klivans and Spielman [KS01]. We deduce the key property for Schubert polynomials to be I-friendly, via the transition formula of Lascoux and Schützenberger [LS85]. The concept of I-friendly bases and the corresponding interpolation algorithm may be of independent interest, since they may be used to apply to other bases of (subrings of) the multivariate polynomial ring, e.g. Grothendieck polynomials and Macdonald polynomials [Las13].

We would like to emphasize a subtle point of Schubert polynomials that is cruicial for our algorithm: for $Y_{\mathbf{v}}$, if the monomial $\mathbf{x}^{\mathbf{u}}$ is in $Y_{\mathbf{v}}$, then $\mathbf{v}$ dominates $\mathbf{u}$ reversely; that is, $v_n \geq u_n$, $v_n + v_{n-1} \geq u_n + u_{n-1}$, $\dots$, $v_n + \dots + v_1 \geq u_n + \dots + u_1$. While by no means a difficult property and known to experts, it is interesting that the only reference we can find is a footnote in Lascoux's book [Las13, pp. 62, footnote 4], so we prove it in Lemma 14. On the other hand, in the literature a weaker property is often mentioned, that is $\mathbf{v}$ is no less than $\mathbf{u}$ in the reverse lexicographic order. However, this order turns out to *not* suffice for the purpose of interpolation.

Comparison with the Barvinok-Fomin algorithm. The underlying structures of the Barvinok-Fomin algorithm and ours are quite similar. There are certain major differences though.

From the mathematical side, note that Barvinok and Fomin used the dominance order of monomials, which corresponds to the use of upper triangular matrix in Section 4.1. On the other hand, we make use of the reverse dominance order, which corresponds to the use of lower traingular matrix in Proposition 18. It is not hard to see that the dominance order could not work for all Schubert polynomials. We also need to upgrade several points (e.g. the computation, and the bounds on coefficients) from Schur polynomials to Schubert polynomials.

From the algorithmic side, both our algorithm and the Barvinok-Fomin algorithm reduce multivariate interpolation to univariate interpolation. Here are two key differences. Firstly, Barvinok and Fomin relied on randomness to obtain a set of linear forms s.t. most of them achieve distinct values for a small set of vectors. We resort to a deterministic construction of Klivans and Spielman for this set, therefore derandomizing the Barvinok-Fomin algorithm. Secondly, our algorithm has a recursive structure as the Barvinok-Fomin algorithm. But in each recursive step, the approaches are different; ours is based on the method of the Klivans-Spielman algorithm. As a consequence, our algorithm does not need to know the bounds on the coefficients in the expansion, while the basic algorithm in [BF97, Sec. 4.1] does. Barvinok and Fomin avoided the dependence on this bound via binary search and probabilistic verification in [BF97, Sec. 4.2]. However, it seems difficult to derandomize this probabilistic verification procedure.

Organization. In Section 2 we present certain preliminaries. In Section 3 we define skew Schubert polynomials and Schubert polynomials, and present the proof for Theorem 2, Corollary 3 and Theorem 4. In Section 4 we define interpolation-friendly bases, and present the interpolation algorithm in such bases. In Section 5 we prove that Schubert polynomials form an I-friendly basis, therefore proving Theorem 1. We remind the reader that skew Schubert polynomials are only studied in Section 3.

2 Preliminaries

Notations. For $n \in \mathbb{N}$, $[n] := \{1, \dots, n\}$. Let $\mathbf{x} = (x_1, \dots, x_n)$ be a tuple of n variables. When no ambiguity, $\mathbf{x}$ may represent the set $\{x_1, \dots, x_n\}$. For $\mathbf{e} = (e_1, \dots, e_n) \in \mathbb{N}^n$, the monomial with exponent $\mathbf{e}$ is $\mathbf{x}^{\mathbf{e}} := x_1^{e_1} \dots x_n^{e_n}$. Given $f \in \mathbb{Z}[\mathbf{x}]$ and $\mathbf{e} \in \mathbb{N}^n$, $\mathrm{Coeff}(\mathbf{e}, f)$ denotes the coefficient of

$\mathbf{x}^{\mathbf{e}}$ in f. $\mathbf{x}^{\mathbf{e}}$ (or $\mathbf{e}$) is in f if $\mathrm{Coeff}(\mathbf{e}, f) \neq 0$, and $E_f := \{\mathbf{e} \in \mathbb{N}^n \mid \mathbf{x}^{\mathbf{e}} \in f\}$. Given two vectors $\mathbf{c} = (c_1, \dots, c_n)$ and $\mathbf{e} = (e_1, \dots, e_n)$ in $\mathbb{Q}^n$, their inner product is $\langle \mathbf{c}, \mathbf{e} \rangle = \sum_{i=1}^n c_i e_i$. Each $\mathbf{c} \in \mathbb{Q}^n$ defines a linear form $\mathbf{c}^\star$, which maps $\mathbf{e} \in \mathbb{Q}^n$ to $\langle \mathbf{c}, \mathbf{e} \rangle$.

Codes and permutations. We call $\mathbf{v} = (v_1, \dots, v_n) \in \mathbb{N}^n$ a *code*. We identify $\mathbf{v}$ with $\mathbf{v}' = (v_1, \dots, v_n, 0, \dots, 0) \in \mathbb{N}^m$, for $m \geq n$. The weight of the code $\mathbf{v}$, denoted by $|\mathbf{v}|$, is $\sum_{i=1}^n v_i$. A code $\mathbf{v} \in \mathbb{N}^n$ is *dominant* if $v_1 \geq v_2 \geq \cdots \geq v_n$. We define a *partition* to be a dominant code, and often represent it using α. $\mathbf{v}$ is *anti-dominant* if $v_1 \leq v_2 \leq \cdots \leq v_k$ and $v_{k+1} = \cdots = v_n = 0$.

For $N \in \mathbb{N}$, S_N is the symmetric group on $[N]$. A permutation $\sigma \in S_N$ is written as $\underline{\sigma(1), \sigma(2), \dots, \sigma(N)}$. We identify σ with $\sigma' = \underline{\sigma(1), \dots, \sigma(N), N+1, \dots, M} \in S_M$, for $M \geq N$. The length of σ, denoted by $|\sigma|$, is the number of inversions of σ. That is, $|\sigma| = |\{(i, j) : i < j; \sigma(i) > \sigma(j)\}|$.

Given a permutation $\sigma \in S_N$, we can associate a code $\mathbf{v} \in \mathbb{N}^n$,[7] by assigning $v_i = |\{j : j > i, \sigma(j) < \sigma(i)\}|$. On the other hand, given a code $\mathbf{v} \in \mathbb{N}^n$ we associate a permutation $\sigma \in S_N (N \geq n)$ as follows. (N will be clear from the construction procedure.) To start, $\sigma(1)$ is assigned as $v_1 + 1$. $\sigma(2)$ is the $(v_2 + 1)$th number, skipping $\sigma(1)$ if necessary (i.e. if $\sigma(1)$ is within the first $(v_2 + 1)$ numbers). $\sigma(k)$ is then the $(v_k + 1)$th number, skipping some of $\sigma(1), \dots, \sigma(k-1)$ if necessary. For example, it can be verified that $\underline{316245}$ gives the code $(2, 0, 3, 0, 0, 0) = (2, 0, 3)$ and vice versa.

Given a code $\mathbf{v}$ its associated permutation is denoted as $\langle \mathbf{v} \rangle$. Conversely, the code of a permutation $\sigma \in S_N$ is denoted as $\mathfrak{c}(\sigma)$. It is clear that $|\sigma| = |\mathbf{v}|$.

Bases. Let R be a (possibly nonproper) subring of the polynomial ring $\mathbb{Z}[\mathbf{x}]$. Suppose M is a basis of R as a $\mathbb{Z}$-module. M is usually indexed by some *index set* Λ, and $M = \{t_\lambda \mid \lambda \in \Lambda\}$. For example $\Lambda = \mathbb{N}^n$ for $R = \mathbb{Z}[\mathbf{x}]$, and $\Lambda = \{\text{partitions in } \mathbb{N}^n\}$ for $R = \{\text{symmetric polynomials}\}$. $f \in R$ can be expressed uniquely as $f = \sum_{\lambda \in \Gamma} a_\lambda t_\lambda$, $a_\lambda \neq 0 \in \mathbb{Z}$, $t_\lambda \in M$, and a finite $\Gamma \subseteq \Lambda$.

A construction of Klivans and Spielman. We present a construction of Klivans and Spielman that is the key to the derandomization here. Given positive integers m, n, and $0 < \epsilon < 1$, let $t = \lceil m^2 n/\epsilon \rceil$. Let d be another positive integer, and fix a prime p larger than t and d. Now define a set of t vectors in $\mathbb{N}^n$ as $\mathrm{KS}(m, n, \epsilon, d, p) := \{\mathbf{c}^{(1)}, \dots, \mathbf{c}^{(t)}\}$ by $\mathbf{c}_i^{(k)} = k^{i-1} \bmod p$, for $i \in [n]$. Note that $\mathbf{c}_i^{(k)} \leq p = O(m^2 n d/\epsilon)$. The main property we need from KS is the following.

Lemma 6 ([KS01, Lemma 3])**.** *Let $\mathbf{e}^{(1)}, \dots, \mathbf{e}^{(m)}$ be m distinct vectors from $\mathbb{N}^n$ with entries in $\{0, 1, \dots, d\}$. Then*

$$\Pr_{k \in [t]}[\langle \mathbf{c}^{(k)}, \mathbf{e}^{(j)} \rangle \textit{ are distinct for } j \in [m]] \geq 1 - m^2 n/t \geq 1 - \epsilon$$

On #P and VNP. The standard definition of #P is as follows: function $f : \cup_{n \in \mathbb{N}} \{0,1\}^n \to \mathbb{Z}$ is in #P, if there exists a polynomial-time Turing machine M and a polynomial p, s.t. for any $x \in \{0,1\}^n$, $f(x) = |\{y \in \{0,1\}^{p(n)} \text{ s.t. } M \text{ accepts } (x, y)\}|$. In the proof of Theorem 2 in Section 3, we find it handy to consider the class of Turing machines that output a nonnegative integer (instead of just accept or reject), and functions $f : \cup_{n \in \mathbb{N}} \{0,1\}^n \to \mathbb{N}$ s.t. there exists a polynomial-time Turing machine M and a polynomial p, s.t. for any $x \in \{0,1\}^n$, $f(x) = \sum_{y \in \{0,1\}^{p(n)}} M(x, y)$.

[7] This is known as the *Lehmer code* of a permutation; see e.g. [Man01, Section 2.1].

As described in [dCSW13], such functions are also in #P, as we can construct a usual Turing machine M', which takes 3 arguments (x, y, z), and $M'(x, y, z)$ accepts if and only if $z < M(x, y)$. Then $\sum_{y,z} M'(x, y, z) = \sum_y M(x, y)$. Note that $z \in \{0, 1\}^{q(n)}$ for some polynomial q as M is polynomial-time.

The reader is referred to [SY10] for basic notions like arithmetic circuits. VP denotes the class of polynomial families $\{f_n\}_{n\in\mathbb{N}}$ s.t. each f_n is a polynomial in $\mathsf{poly}(n)$ variables, of $\mathsf{poly}(n)$ degree, and can be computed by an arithmetic circuit of size $\mathsf{poly}(n)$. VNP is the class of polynomials $\{g_n\}_{n\in\mathbb{N}}$ s.t. $g_n(x_1, \dots, x_n) = \sum_{(c_1,\dots,c_m)\in\{0,1\}^m} f_n(x_1, \dots, x_n, c_1, \dots, c_m)$ where $m = \mathsf{poly}(n)$, and $\{f_n\}_{n\in\mathbb{N}}$ is in VP. Valiant's criterion is useful to put polynomial families in VNP.

Theorem 7 (Valiant's criterion, [Val79])**.** *Suppose $\phi : \{0, 1\}^\star \to \mathbb{N}$ is a function in #P/*poly*. Then the polynomial family $\{f_n\}_{n\in\mathbb{N}}$ defined by $f_n = \sum_{\mathbf{e}\in\{0,1\}^n} \phi(\mathbf{e})\mathbf{x}^{\mathbf{e}}$ is in* VNP*.*

3 Skew Schubert polynomials in #P and VNP

In this section we first define skew Schubert polynomials via the labeled Bruhat order as in [LS03]. We also indicate how Schubert polynomials form a special case of skew Schubert polynomials. We then put these polynomials, and therefore Schubert polynomials, in #P and VNP. Also note that it is more convenient to work with permutations instead of codes in this section.

Definition of skew Schubert polynomials. The Bruhat order on permutations in S_N is defined by its covers: for $\sigma, \pi \in S_N$, $\sigma \dot{<} \pi$ if (i) $\sigma^{-1}\pi$ is a transposition τ_{ik} for $i < k$, and (ii) $|\sigma| + 1 = |\pi|$. Assuming (i), condition (ii) is equivalent to:

(a) $\sigma(i) < \sigma(k)$;

(b) for any j such that $i < j < k$, either $\sigma(j) > \sigma(k)$, or $\sigma(j) < \sigma(i)$.

This is because π gets an inversion added due to the transposition τ_{ik}. So in no position between i and k can σ take a value between $\sigma(i)$ and $\sigma(k)$. Else, the number of inversions in π will change by more than 1. Taking the transitive closure gives the Bruhat order $(\le)$. The maximal element in Bruhat order is $\pi_0 = \underline{N, N-1, \dots, 1}$, whose code is $\mathbf{d} = (N-1, N-2, \dots, 1)$.

The *labeled Bruhat order* is the key to the definition of skew Schubert polynomials. While naming it as an order, it is actually a directed graph with multiple labeled edges, with the vertices being the permutations in S_N. For $\sigma \dot{<} \pi$ s.t. $\sigma^{-1}\pi = \tau_{st}$, $s \le j < t$ and $b = \sigma(s) = \pi(t)$, add a labeled direct edge as $\sigma \xrightarrow{(j,b)} \pi$. That is, for each $\sigma \dot{<} \pi$, there are $t - s$ edges between them.

For any saturated chain C in this graph, we associate a monomial $\mathbf{x}^{\mathbf{e}(C)}$, where $\mathbf{e}(C) = (e_1, \dots, e_{N-1})$, and e_i counts the number of i appearing as the first coordinate of a label in C. A chain

$$\sigma_0 \xrightarrow{(j_1,b_1)} \sigma_1 \xrightarrow{(j_2,b_2)} \dots \xrightarrow{(j_m,b_m)} \sigma_m$$

is *increasing* if its sequence of labels is increasing in the lexicographic order on pairs of integers.

Now we arrive at the definition of skew Schubert polynomials.

Definition 8. *Let $\mathbf{d}$ and π_0 be as above. Given two permutations σ and π, s.t. σ is no larger than π in the Bruhat order ($\sigma \le \pi$), the* skew Schubert polynomial

$$Y_{\pi/\sigma}(\mathbf{x}) := \sum_C \mathbf{x}^{\mathbf{d}}/\mathbf{x}^{\mathbf{e}(C)}, \tag{1}$$

summing over all increasing chains in the labeled Bruhat order from σ to π. The Schubert polynomial $Y_\sigma := Y_{\pi_0/\sigma}$.

Now in the increasing chain (or any chain in the labeled Bruhat order), each edge increases $|\sigma|$ by 1. So the number of edges in an increasing chain from σ to π is $|\pi| - |\sigma|$.

Skew Schubert polynomials in #P and VNP**.** Before describing the #P algorithm for skew Schubert polynomials, we note the following. First, by the correspondence between codes and permutations, from a code $\mathbf{v} \in \mathbb{N}^n$ we can compute $\langle \mathbf{v} \rangle \in S_N$ in time polynomial in $|\mathbf{v}|$. Also, we have $N = \max_{i\in[n]}\{v_i + i\} \leq |\mathbf{v}_i| + n \leq |\mathbf{v}_0| + n$. Second, the length of the path from σ to π is $|\pi| - |\sigma|$.

To start with let us see how Corollary 3 follows from Theorem 2.

Proof of Corollary 3. Given $\mathbf{v}$, we can compute $\langle \mathbf{v} \rangle \in S_N$. Note that $N \leq |\mathbf{v}| + n$. Then form $\mathbf{w} = (N-1, N-2, \dots, 1)$. Invoke Theorem 2 with $(\mathbf{v}, \mathbf{w}, \mathbf{a})$. □

Proof of Theorem 2. Consider the following Turing machine M: the input to M are (1) codes $\mathbf{v}, \mathbf{w} \in \mathbb{N}^m$ in unary; (2) $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{N}^n$ in binary; and (3) a sequence $\mathbf{s}$ of triplets of integers $(s_i, j_i, t_i) \in [N] \times [N] \times [N]$ where $s_i \leq j_i < t_i$, $N = |\mathbf{w}| + m$, and $i \in [\ell]$, $\ell = |\mathbf{w}| - |\mathbf{v}|$. (Note that all the conditions on $\mathbf{s}$ can be checked efficiently.) The output of M is a nonnegative integer.

Given the input $(\mathbf{v}, \mathbf{w}, \mathbf{a}, \mathbf{s})$, M uses $\mathbf{s}$ as a guide to compute an increasing chain from $\langle \mathbf{v} \rangle$ to $\langle \mathbf{w} \rangle$ in the labeled Bruhat order of S_N. Let $\sigma_0 = \langle \mathbf{v} \rangle$. Suppose the chain to be constructed is $\sigma_0 \xrightarrow{(j_1,b_1)} \sigma_1 \xrightarrow{(j_2,b_2)} \dots \xrightarrow{(j_\ell,b_\ell)} \sigma_\ell$. At step i, $0 \leq i < \ell$, M also maintains an exponent vector $\mathbf{e}_i \in \mathbb{N}^{N-1}$, and the label (j_i, b_i).

To start, M sets $\mathbf{e}_0 = (0, \dots, 0)$, and $(j_0, b_0) = (0, 0)$ (lexicographically minimal). Then when the step $i-1$ finishes, M maintains $\mathbf{e}_{i-1}$, σ_{i-1}, and (j_{i-1}, b_{i-1}). Then at step i, based on (s_i, j_i, t_i), M performs the following. First, it computes $\sigma_i = \sigma_{i-1}\tau_{s_i t_i}$, and checks whether $|\sigma_{i-1}| + 1 = |\sigma_i|$: if equal, then continue; otherwise, return 0 (not a valid chain). Second, it sets $\mathbf{e}_i$ by adding 1 to the j_ith component of $\mathbf{e}_{i-1}$, and keeping the other components same. Third, it computes $b_i = \sigma_i(t_i)$, and checks whether (j_i, b_i) is larger than (j_{i-1}, b_{i-1}) in the lexicographic order: if it is larger, then continue; otherwise, return 0 (not a valid chain).

When the ℓth step finishes, M obtains σ_ℓ and $\mathbf{e}_\ell$. It first checks whether $\sigma_\ell = \langle \mathbf{w} \rangle$: if equal, then continue. Otherwise, return 0 (not a valid chain). M then computes $\mathbf{a}^{\mathbf{d}}/\mathbf{a}^{\mathbf{e}_\ell}$ as the output.

This finishes the description of M. Clearly M runs in time polynomial in the input size. M terminates within ℓ steps where $\ell = |\mathbf{w}| - |\mathbf{v}|$, and recall that $\mathbf{w}$ and $\mathbf{v}$ are given in unary.

Finally note that $Y_{\mathbf{w}/\mathbf{v}}(\mathbf{a})$ is equal to $\sum_{\mathbf{s}} M(\mathbf{v}, \mathbf{w}, \mathbf{a}, \mathbf{s})$, where $\mathbf{s}$ runs over all sequences of triplets of indices as described at the beginning of the proof. By the discussion of #P at the end of Section 2, this puts the evaluation of $Y_{\mathbf{w}/\mathbf{v}}$ in #P. □

Proof of Theorem 4. Let us outline the proof of Theorem 4, which basically follows from the proof of Theorem 2. By Valiant's criterion (Theorem 7), to put $Y_{\mathbf{w}/\mathbf{v}}$ in VNP, it suffices to show that the coefficient of any monomial in $Y_{\mathbf{w}/\mathbf{v}}$ is in #P. Therefore we consider the following Turing machine M', which is modified from the Turing machine M in the proof of Theorem 2 as follows. Firstly, M' takes input $(\mathbf{v}, \mathbf{w}, \mathbf{a}, \mathbf{s})$, where $\mathbf{a}$ is thought of as an exponent vector, and is given in unary. Second, in the last step, M' checks whether $\mathbf{d} - \mathbf{e}_\ell$ equals $\mathbf{a}$ or not. If equal, then output 1. Otherwise output 0. It is clear that this gives a #P algorithm to compute the coefficient of $\mathbf{x}^{\mathbf{a}}$. The only small problem here is that in the literature ([Bür00, Prop. 2.20], [Koi05, Thm. 2.3]) we can find, Valiant's criterion is only stated for multilinear polynomials.[8] But it only takes a little more effort to overcome this; essentially, this is because the degree is assumed to be polynomially bounded, so $\mathbf{a}$ can be given in unary. First, note that N (as in the beginning of the proof of Theorem 2)is an upper bound on the individual degree for each x_i. Then introduce N copies for each variable x_i. Since $\mathbf{a} = (a_1, \dots, a_n)$ is given in unary, we can assume w.l.o.g. that each a_i is an N-bit string $(a_{i,1}, \dots, a_{i,N})$, and if $a_i = k$, the first k bits are set to 1, and the rest 0. (These conditions are easy to enforce in the definition of M'.) We then use $a_{i,j}$ to control whether we have the jth copy of x_i, or 1, using the formula $a_{i,j}x_i + 1 - a_{i,j}$. After this slight modification, the Valiant's criterion applies, and the proof is concluded. □

4 Sparse interpolation in an interpolation-friendly basis

4.1 Interpolation-friendly bases

Let $M = \{t_\lambda \mid \lambda \in \Lambda\}$ be a basis of a $\mathbb{Z}$-module $R \subseteq \mathbb{Z}[x_1, \dots, x_n]$, indexed by $\lambda \in \Lambda$. Given a function $K : \Lambda \times \mathbb{N} \to \mathbb{R}^+$, M is called K*-bounded*, if $\forall t_\lambda \in M$, the absolute values of the coefficients in the monomial expansion of $t_\lambda \in M$, $t_\lambda \in \mathbb{Z}[x_1, \dots, x_n]$ are bounded by $K(\lambda, n)$.[9]

For a $(0, 1)$, nonsingular matrix A, $L_A := \{\mathbf{c}^\star \in (\mathbb{Q}^n)^\star \mid \exists \mathbf{c}' \in (\mathbb{Z}^+)^n, \mathbf{c} = A\mathbf{c}'\}$. Note that $\mathbf{c}'$ is a vector with positive integer components. Also recall that for $\mathbf{c} \in \mathbb{Q}^n$, $\mathbf{c}^\star$ denotes the linear form determined by $\mathbf{c}$. M is L_A*-compatible*, if for every $t_\lambda \in M$, we can associate an exponent $\mathbf{e}_\lambda$ s.t. (1) for any $\mathbf{c}^\star \in L_A$, $\mathbf{c}^\star$ achieves the maximum uniquely at $\mathbf{e}_\lambda$ over $E_{t_\lambda} = \{\mathbf{e} \in \mathbb{N}^n \mid \mathbf{x}^\mathbf{e} \in t_\lambda\}$; (2) $\mathbf{e}_\lambda \neq \mathbf{e}_{\lambda'}$ for $\lambda \neq \lambda'$; (3) the coefficient of $\mathbf{x}^{\mathbf{e}_\lambda}$ in t_λ is 1. $\mathbf{x}^{\mathbf{e}_\lambda}$ (resp. $\mathbf{e}_\lambda$) is called the *leading monomial* (resp. *leading exponent*) of t_λ w.r.t. L_A. By the conditions (1) and (2), the leading monomials are distinct across M, and for each t_λ, the leading monomial is unique. We assume that from λ it is easy to compute $\mathbf{e}_\lambda$, and vice versa. In fact, for Schubert polynomials, λ is from $\Lambda = \mathbb{Z}^n$, and $\mathbf{e} = \lambda$.

Combining the above two definitions, we say M is (K, L_A)-interpolation-friendly, if (1) M is K-bounded; (2) M is L_A-compatible. We also call it (K, L_A)-friendly for short, or I-friendly when K and L_A are understood from the context.

- A trivial example is the monomial basis for $\mathbb{Z}[\mathbf{x}]$, where $K = 1$, A is the identity transformation, and a leading monomial for $\mathbf{x}^\mathbf{e}$ is just itself;
- For symmetric polynomials, the basis of Schur polynomials (indexed by partitions α) is (K, L_A)-friendly, where (1) $K(\alpha, n) = \sqrt{|\alpha|!}$ by Proposition 11, (2) $A = (r_{i,j})_{i,j\in[n]}$ where $r_{i,j} = 0$ if $i > j$, and 1 otherwise, by the fact that every exponent vector in s_α is dominated by α ([BF97, Sec. 2.2]). (A is the upper triangular matrix with 1's on the diagonal and above.) The associated leading monomial for s_α is $\mathbf{x}^\alpha$. In retrospect, the fact that Schur polynomials form an I-friendly basis is the mathematical support of the Barvinok-Fomin algorithm [BF97].

[8] It is well-known though that Valiant's criterion works for even non-multilinear polynomials. However, the only reference we are aware of is [Sap16, pp. 10, Footnote 1], where no details are provided. Therefore, we describe the procedure to overcome this for completeness.

[9] Note that while Λ depends on n already, we feel that it is clearer to explicitly designate n as a parameter of K, as seen e.g. in Proposition 17.

4.2 Sparse interpolation in an interpolation-friendly basis

In this section we perform deterministic sparse polynomial interpolation in an interpolation-friendly basis. The idea is to combine the structure of the Barvinok-Fomin algorithm with some ingredients from the Klivans-Spielman algorithm.

We first briefly review the idea of the Klivans-Spielman algorithm [KS01, Section 6.3]. Suppose we want to interpolate $f \in \mathbb{Z}[x_1, \ldots, x_n]$ of degree d with m monomials. Their algorithm makes use of the map

$$\phi_{\mathbf{c}}(x_1, \ldots, x_n) = (y^{c_1}, \ldots, y^{c_n}) \tag{2}$$

where $\mathbf{c} = (c_1, \ldots, c_n) \in (\mathbb{Z}^+)^n$. If $\mathbf{c}$ satisfies the property: $\forall \mathbf{e} \neq \mathbf{e}' \in f, \langle \mathbf{c}, \mathbf{e} \rangle \neq \langle \mathbf{c}, \mathbf{e}' \rangle$, then we can reduce interpolation of multivariate polynomials to interpolation of univariate polynomials: first apply the univariate polynomial interpolation (based on the Vandermonde matrix) to get a set of coefficients. Then to recover the exponents, modify $\phi_{\mathbf{c}}$ as

$$\phi'_{\mathbf{c}}(x_1, \ldots, x_n) = (p_1 y^{c_1}, \ldots, p_n y^{c_n}), \tag{3}$$

where p_i's are distinct primes, and get another set of coefficients. Comparing the two sets of coefficients we can compute the exponents. Note that the components of $\mathbf{c}$ need to be small for the univariate interpolation to be efficient. To obtain such a $\mathbf{c}$, Klivans and Spielman exhibit a small – polynomial in n, m, d and an error probability $\epsilon \in (0, 1)$ – set of test vectors $\mathbf{c}$ s.t. with probability $1 - \epsilon$, a vector from this set satisfies the above property. Furthermore, the components of these vectors are bounded by $O(m^2 nd/\epsilon)$. Their construction was reviewed in Lemma 6, Section 2.

Now suppose M is a (K, L_A)-friendly basis for $R \subseteq \mathbb{Q}[\mathbf{x}]$, and we want to recover $f = \sum_{\lambda \in \Gamma} a_\lambda t_\lambda$ of degree $\leq d$, and $|\Gamma| = m$. To apply the above idea to an arbitrary I-friendly basis M, the natural strategy is to extract the leading monomials w.r.t. L_A. However, as each basis polynomial can be quite complicated, there are many other non-leading monomials which may interfere with the leading ones. Specifically, we need to explain the following:

(1) Whether extremely large coefficients appear after the map $\phi_{\mathbf{c}}$, therefore causing the univariate interpolation procedure to be inefficient?

(2) Whether some leading monomials are preserved after the map $\phi_{\mathbf{c}}$? (That is, will the image of every leading monomial under $\phi_{\mathbf{c}}$ be cancelled by non-leading monomials?)

It is immediate to realize that I-friendly bases are designed to overcome the above issues. (1) is easy: by the K-bounded property, for any monomial $\mathbf{x}^{\mathbf{u}}$ in f, the absolute value of Coeff$(\mathbf{u}, f)$ is bounded by $K \cdot (\sum_\lambda |a_\lambda|)$. Therefore, the coefficients of the image of f under $\phi_{\mathbf{c}}$ are bounded by $O\big(\binom{n+d}{d} \cdot K \cdot (\sum_\lambda |a_\lambda|)\big)$. (2) is not hard to overcome either; see the proof of Theorem 9 below. These properties are used implicitly in the Barvinok-Fomin algorithm.

There is one final note: if, unlike in the monomial basis case, the procedure cannot produce all leading monomials at one shot, we may need to get one t_λ and its coefficient, subtract that off, and recurse. This requires us to compute t_λ's efficiently. As this is the property of t_λ, not directly related to the interpolation problem, we assume an oracle which takes an index $\lambda \in \Lambda$ and an input $\mathbf{a} \in \mathbb{N}^n$, and returns $t_\lambda(\mathbf{a})$.

Theorem 9. *Let $M = \{t_\lambda \mid \lambda \in \Lambda\}$ be a (K, L_A)-friendly basis for $R \subseteq \mathbb{Z}[\mathbf{x}]$, $K = K(\lambda, n)$, A a $(0, 1)$ invertible matrix. Given an access to an oracle $\mathcal{O} = \mathcal{O}(\lambda, \mathbf{a})$ that computes basis polynomial $t_\lambda(\mathbf{a})$ for $\mathbf{a} \in \mathbb{N}^n$, there exists a deterministic algorithm that, given a black box containing $f =$*

$\sum_{\lambda\in\Gamma} a_\lambda t_\lambda$, $a_\lambda \neq 0 \in \mathbb{Z}$ *with the promise that* $\deg(f) \leq d$ *and* $|\Gamma| \leq m$, *computes such an expansion of* f *in time* $\mathsf{poly}(n, d, m, \log(\sum_\lambda |a_\lambda|), \log K)$.

Proof. We first present the algorithm. Recall the maps $\phi_{\mathbf{c}}$ and $\phi'_{\mathbf{c}}$ defined in Equation (2) and Equation (3).

Input: A black box $\mathcal{B}$ containing $f \in R \subseteq \mathbb{Z}[x_1, \dots, x_n]$ with the promises: (1) $\deg(f) \leq d$; (2) f has $\leq m$ terms in the M-basis. An oracle $\mathcal{O} = \mathcal{O}(\lambda, \mathbf{a})$ computing $t_\lambda(\mathbf{a})$ for $\mathbf{a} \in \mathbb{N}^n$.

Output: The expansion $f = \sum_{\lambda\in\Gamma} a_\lambda t_\lambda$.

Algorithm:

```
1. By Lemma 6, construct the Klivans-Spielman set KS = KS($m, n, 1/3, nd, p$).
2. For every vector $\mathbf{c}$ in KS, do:
   (a) $f_{\mathbf{c}} \leftarrow 0$. $i \leftarrow 0$. $\mathbf{d} \leftarrow A\mathbf{c}$.
   (b) While $i < m$, do:
       i.   Apply the map $\phi_{\mathbf{d}}$ to $\mathcal{B} - f_{\mathbf{c}}$ (with the help of $\mathcal{O}$), and use the univariate
            interpolation algorithm to obtain $g(y) = \sum_{i=0}^{k} b_i y^i$. If $g(x) \equiv 0$, break.
       ii.  Apply the map $\phi'_{\mathbf{d}}$ to $\mathcal{B} - f_{\mathbf{c}}$ (with the help of $\mathcal{O}$), and use the univariate
            interpolation algorithm to obtain $g'(y) = \sum_{i=0}^{k} b'_i y^i$.
       iii. From $b_k$ and $b'_k$, compute the corresponding monomial $\mathbf{x}^{\mathbf{e}}$, and its coefficient $a_{\mathbf{e}}$.
            From $\mathbf{x}^{\mathbf{e}}$, compute the corresponding label $\nu \in \Lambda$, and set $a_\nu \leftarrow a_{\mathbf{e}}$.
       iv.  $f_{\mathbf{c}} \leftarrow f_{\mathbf{c}} + a_\nu t_\nu$. $i \leftarrow i + 1$.
3. Take the majority of $f_{\mathbf{c}}$ over $\mathbf{c}$ and output it.
```

We prove the correctness of the above algorithm. As before, let $\mathbf{e}_\lambda$ be the leading vector of t_λ w.r.t. L_A. Note that as A^T is $(0,1)$-matrix, entries in the vector $A^T\mathbf{e}_\lambda$ are in $\{0, 1, \dots, nd\}$. By the property of KS$(m, n, 1/3, nd, p)$, no less than $2/3$ fraction of the vectors from $\mathbf{c} \in$ KS satisfy that $\langle \mathbf{c}, A^T\mathbf{e}_\lambda\rangle$ are distinct over $\lambda \in \Gamma$; call these vectors "distinguishing." We shall show that for any distinguishing vector, the algorithm outputs the correct expansion, so step (3) would succeed.

Fix a distinguishing vector $\mathbf{c}$. As $\mathbf{e}_\lambda \neq \mathbf{e}_{\lambda'}$ for $\lambda \neq \lambda'$ and A is invertible, there exists a unique $\nu \in \Lambda$ s.t. $\mathbf{e}_\nu$ achieves the maximum at $\langle \mathbf{c}, A^T\mathbf{e}_\lambda\rangle$ over $\lambda \in \Gamma$. As $\langle \mathbf{c}, A^T\mathbf{e}_\lambda\rangle = \langle A\mathbf{c}, \mathbf{e}_\lambda\rangle = \langle \mathbf{d}, \mathbf{e}_\lambda\rangle$, by the definition of M being L_A-compatible, we know that within each t_λ, $\mathbf{d}^\star$ achieves the unique maximum at $\mathbf{e}_\lambda$ over E_{t_λ}, the set of all exponent vecors in t_λ. Thus over all $\mathbf{e} \in f$, $\mathbf{d}^\star$ achieves the unique maximum at $\mathbf{e}_\nu$. This means that $y^{\langle \mathbf{d}, \mathbf{e}_\nu\rangle}$ is the monomial in $g(y)$ of maximum degree, and could not be affected by other terms. As Coeff$(\mathbf{e}_\nu, t_\nu) = 1$, Coeff$(\mathbf{e}_\nu, f)$ is just the coefficient of t_ν in the expansion of f. So we have justified that from Step (2.b.i) to (2.b.iii), the algorithm extracts the monomial of maximum degree in $g(y)$, computes the corresponding coefficient and exponent in f, and interprets as a term $a_\lambda t_\lambda$.

To continue computing other terms, we just need to note that $\mathbf{c}$ is still distinguishing w.r.t. ($f-$ some of the terms within f). This justifies Step (2.b.iv).

To analyze the running time, the FOR-loop in Step (2) and the WHILE-loop in Step (2.b) take $O(m^2 n)$ and m rounds, respectively. In the univariate polynomial interpolation step, as the components in $\mathbf{c}$ are bounded by $O(m^2 n \cdot nd)$ and A is $(0,1)$, the components in $\mathbf{d}$ are bounded by $O(m^2 n^3 d)$. It follows that $k = \deg(g) = \langle \mathbf{d}, \mathbf{e}_\nu\rangle = O(m^2 n^3 d^2)$. By the K-bounded property, the coefficients of $g(y)$ are of magnitude $O\big(\binom{n+d}{d} \cdot K \cdot (\sum_\lambda |a_\lambda|)\big)$. So the running time for the univariate interpolation step, and therefore for the whole algorithm, is $\mathsf{poly}(m, n, d, \log(\sum_\lambda |a_\lambda|), \log K)$. □

5 Schubert polynomials form an interpolation-friendly basis

In this section, our ultimate goal is to prove Proposition 17 and Proposition 18, which establish that Schubert polynomials form an I-friendly basis. The main theorem Theorem 1 follows immediately. For this, we need to review some properties of Schur polynomials, the definition of Schubert polynomials via divided differences, and the transition formula [LS85]. The transition formula is the main technical tool to deduce the properties of Schubert polynomials we shall need for Proposition 17 and Proposition 18. These include Lemma 14 which helps us to find the matrix A needed for the L_A-compatible property, and an alterative proof for Schubert polynomial in #P.

Schur polynomials. For a positive integer ℓ, the complete symmetric polynomial $h_\ell(\mathbf{x}) \in \mathbb{Z}[\mathbf{x}]$ is the sum over all monomials of degree ℓ, with coefficient 1 for every monomial. We also define $h_0(\mathbf{x}) = 1$, and $h_\ell(\mathbf{x}) = 0$ for any $\ell < 0$. For a partition $\alpha = (\alpha_1, \dots, \alpha_n)$ in $\mathbb{N}^n$, the Schur polynomial $\mathrm{s}_\alpha(\mathbf{x})$ in $\mathbb{Z}[\mathbf{x}]$ can be defined by the Jacobi-Trudi formula as $\mathrm{s}_\alpha(\mathbf{x}) = \det[h_{\alpha_i - i + j}(\mathbf{x})]_{i,j\in[n]}$. Note that $\deg(\mathrm{s}_\alpha(\mathbf{x})) = |\alpha|$. Via this determinantal expression, we have

Proposition 10 ([BF97, Sec. 2.4])**.** *For $\mathbf{a} \in \mathbb{Z}^n$, $\mathrm{s}_\alpha(\mathbf{a})$ can be computed using $O(|\alpha|^2 \cdot n + n^3)$ arithmetic operations, and the bit lengths of intermediate numbers are polynomial in those of $\mathbf{a}$.*

Littlewood's theorem shows Schur polynomials have positive coefficients. We also need the following bound on coefficients – the Kostka numbers – in $\mathrm{s}_\alpha(\mathbf{x})$.

Proposition 11 ([BF97, Sec. 2.2])**.** *For any $\mathbf{e} \in \mathbb{N}^n$, $0 \le \mathrm{Coeff}(\mathbf{e}, \mathrm{s}_\alpha(\mathbf{x})) \le \sqrt{|\alpha|!}$.*

Definition of Schubert polynomials via divided differences. We follow the approach in [Las03, Las08]. For $i \in [n-1]$, let χ_i be the switching operator on $\mathbb{Z}[x_1, \dots, x_n]$:

$$f^{\chi_i}(x_1, \dots, x_i, x_{i+1}, \dots, x_n) := f(x_1, \dots, x_{i+1}, x_i, \dots, x_n).$$

Then the divided difference operator ∂_i on $\mathbb{Z}[x_1, \dots, x_n]$ is $\partial_i(f) := \frac{f - f^{\chi_i}}{x_i - x_{i+1}}$.

Definition 12. *For $\mathbf{v} = (v_1, \dots, v_n) \in \mathbb{N}^n$, the Schubert polynomial $Y_\mathbf{v} \in \mathbb{Z}[x_1, \dots, x_n]$ is defined recursively as follows:*

1. *If $\mathbf{v}$ is dominant, then $Y_\mathbf{v} = x_1^{v_1} x_2^{v_2} \dots x_n^{v_n}$.*

2. *If $v_i > v_{i+1}$, then $Y_{\mathbf{v}'} = \partial_i Y_\mathbf{v}$ where $\mathbf{v}' = (v_1, \dots, v_{i+1}, v_i - 1, \dots, v_n)$.*

It is not hard to see that this defines $Y_\mathbf{v}$ for any $\mathbf{v}$. We list some basic facts about Schubert polynomials.

Fact 13 ([Man01])**.**

1. *If $\mathbf{v} = (v_1, \dots, v_n)$ is anti-dominant with k being the last nonzero index, $Y_\mathbf{v}$ equals the Schur polynomial $\mathrm{s}_\alpha(x_1, \dots, x_k)$ where $\alpha = (v_k, \dots, v_1)$.*

2. *As a special case of (1), if $\mathbf{v} = (0, \dots, 0, w, 0, \dots 0)$ where w is at the kth position, then $Y_\mathbf{v}(\mathbf{x})$ is the complete homogeneous symmetric polynomial $h_w(x_1, \dots, x_k)$.*

3. *If $\mathbf{v} = [v_1, \dots, v_k, 0, \dots, 0] \in \mathbb{N}^n$, then $Y_\mathbf{v} \in \mathbb{Z}[x_1, \dots, x_k]$.*

The transition formula and its applications. Given a code $\mathbf{v} \in \mathbb{N}^n$, let k be the largest t s.t. v_t is nonzero, and $\mathbf{v}' = [v_1, \ldots, v_k - 1, 0, \ldots, 0]$. For convenience let $\sigma = \langle \mathbf{v}' \rangle$. Then the transition formula of Lascoux and Schützenberger [LS85] is:

$$Y_{\mathbf{v}} = x_k Y_{\mathbf{v}'} + \sum_{\mathbf{u}} Y_{\mathbf{u}},$$

where $\mathbf{u} \in \mathbb{N}^n$ satisfies that: (i) $\langle \mathbf{u} \rangle \sigma^{-1}$ is a transposition τ_{ik} for $i < k$; (ii) $|\mathbf{u}| = |\mathbf{v}|$. Assuming (i), condition (ii) is equivalent to:

$$\text{(a) } \sigma(i) < \sigma(k); \text{(b) for any } j \text{ s.t. } i < j < k, \qquad \text{either } \sigma(j) > \sigma(k), \text{ or } \sigma(j) < \sigma(i). \tag{4}$$

Let $\Psi_{\mathbf{v}}$ be the set of codes with weight $|\mathbf{v}|$ appearing in the transition formula for $\mathbf{v}$, and $\Phi_{\mathbf{v}} = \Psi_{\mathbf{v}} \cup \{\mathbf{v}'\}$. Any $\mathbf{u} \in \Psi_{\mathbf{v}}$ is uniquely determined by the transposition τ_{ik}, therefore, by some $i \in [k-1]$.

The transition formula yields the following simple, yet rarely mentioned[10] property of Schubert polynomials. This is the key to show that the Schubert basis is L_A-compatible for some appropriate A. For completeness we include a proof here.

Given $\mathbf{v}$ and $\mathbf{u}$ in $\mathbb{N}^n$, $\mathbf{v}$ dominates $\mathbf{u}$ reversely, denoted as $\mathbf{v} \triangleright \mathbf{u}$, if $v_n \geq u_n$, $v_n + v_{n-1} \geq u_n + u_{n-1}$, $\ldots$, $v_n + \cdots + v_2 \geq u_n + \cdots + u_2$, $v_n + \cdots + v_1 = u_n + \cdots + u_1$.

Lemma 14. *For $\mathbf{u} \in \mathbb{N}^n$, if $\mathbf{x}^{\mathbf{u}}$ is in $Y_{\mathbf{v}}$, then $\mathbf{v} \triangleright \mathbf{u}$. Furthermore,* $\mathrm{Coeff}(\mathbf{v}, Y_{\mathbf{v}}) = 1$.

Proof. We first induct on the weight. When the weight is 1, the claim holds trivially. Assume the claim holds for weight $\leq w$, and consider $\mathbf{v} \in \mathbb{N}^n$ with $|\mathbf{v}| = w + 1$. We now induct on the reverse dominance order, from small to large. The smallest one with weight $w + 1$ is $[w + 1, 0, \ldots, 0]$. As $Y_{[w+1,0,\ldots,0]} = x_1^{w+1}$, the claim holds.

For the induction step, we make use of the transition formula. Suppose k is the largest i s.t. v_i is nonzero, $\mathbf{v}' = [v_1, \ldots, v_k - 1, 0, \ldots, 0]$, and $\sigma = \langle \mathbf{v}' \rangle$. Then by the transition formula, $Y_{\mathbf{v}} = x_k Y_{\mathbf{v}'} + \sum_{\mathbf{u}} Y_{\mathbf{u}}$, where $\mathbf{u} \in \mathbb{N}^n$ satisfies that: (i) $\langle \mathbf{u} \rangle \sigma^{-1}$ is a transposition τ_{ik} for $i < k$; (ii) $|\mathbf{u}| = |\mathbf{v}|$. Assuming (i), the condition (ii) is equivalent to that: (a) $\sigma(i) < \sigma(k)$; (b) for any j s.t. $i < j < k$, either $\sigma(j) > \sigma(k)$, or $\sigma(j) < \sigma(i)$. Thus $\mathbf{u}$ and $\mathbf{v}'$ can only differ at positions i and k, and $u_i > v'_i = v_i$, $u_k \leq v'_k < v_k$. It follows that $\mathbf{v} \triangleright \mathbf{u}$, and it is clear that $\mathbf{v} \neq \mathbf{u}$. By the induction hypothesis on $|\mathbf{v}|$, each monomial in $Y_{\mathbf{v}'}$ is reverse dominated by $\mathbf{v}'$, and $\mathrm{Coeff}(\mathbf{v}', Y_{\mathbf{v}'}) = 1$. As $Y_{\mathbf{v}'}$ depends only on $x_1, \ldots, x_k$ by 13 (3), each monomial in $x_k Y_{\mathbf{v}'}$ is reverse dominated by $\mathbf{v}$. By the induction hypothesis on the reverse dominance order, every monomial in $Y_{\mathbf{u}}$ is reverse dominated by $\mathbf{u}$, thus is reverse dominated by $\mathbf{v}$ and cannot be equal to $\mathbf{v}$. Thus $\mathrm{Coeff}(\mathbf{v}, Y_{\mathbf{v}}) = 1$, which is from $x_k Y_{\mathbf{v}'}$. This finishes the induction step. □

We then deduce another property of Schubert polynomials from the transition formula. Starting with $\mathbf{v}_0$, we can form a chain of transitions $\mathbf{v}_0 \to \mathbf{v}_1 \to \mathbf{v}_2 \to \cdots \to \mathbf{v}_i \to \ldots$ where $\mathbf{v}_i \in \Psi_{\mathbf{v}_{i-1}}$. The following lemma shows that long enough transitions lead to anti-dominant codes.

Lemma 15 ([LS85, Lemma 3.11])**.** *Let $\mathbf{v}_0 \to \mathbf{v}_1 \to \cdots \to \mathbf{v}_\ell$ be a sequence of codes in $\mathbb{N}^n$, s.t. $\mathbf{v}_i \in \Psi_{\mathbf{v}_{i-1}}$, $i \in [\ell]$. If none of $\mathbf{v}_i$'s are anti-dominant, then $\ell \leq n \cdot |\mathbf{v}_0|$.*

[10] The only reference we know of is in [Las13, pp. 62, Footnote 4].

Based on Lemma 15, we have the following corollary. Recall that for $\mathbf{v} \in \mathbb{N}^n$, $\Phi_{\mathbf{v}}$ is the collection of codes (not necessarily of weight $|\mathbf{v}|$) in the transition formula for $\mathbf{v}$.

Corollary 16. *Let $\mathbf{v}_0 \to \mathbf{v}_1 \to \cdots \to \mathbf{v}_\ell$ be a sequence of codes in $\mathbb{N}^n$, s.t. $\mathbf{v}_i \in \Phi_{\mathbf{v}_{i-1}}$, $i \in [\ell]$. If none of $\mathbf{v}_i$'s are anti-dominant, then $\ell \leq n \cdot (|\mathbf{v}_0|^2 + |\mathbf{v}_0|)$.*

Proof. For $w \in [|\mathbf{v}_0|]$, let i_w be the last index i in $[\ell]$ s.t. $\mathbf{v}_i$ is of weight w. As none of $\mathbf{v}_i$'s are anti-dominant, by Lemma 15, $i_{w-1} - i_w \leq n \cdot w + 1 \leq n \cdot |\mathbf{v}_0| + 1$. The result then follows. □

In fact, by following the proof of Lemma 15 as in [LS85], it is not hard to show that in Corollary 16 the same bound as in Lemma 15, namely $\ell \leq n \cdot |\mathbf{v}_0|$, holds.

From Corollary 16, a #P algorithm for Schubert polynomials can also be derived.

An alternative proof of Corollary 3. Consider the following Turing machine M: the input to M is a code $\mathbf{v} \in \mathbb{N}^n$ in unary, a point $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{N}^n$ in binary, and a sequence $\mathbf{s}$ of pairs of indices $(s_i, t_i) \in [n] \times [n]$, $s_i \leq t_i$, and $i \in [\ell]$ where $\ell := n \cdot (|\mathbf{v}|^2 + |\mathbf{v}|)$. The output of M is a nonnegative integer. Given the input $(\mathbf{v}, \mathbf{a}, \mathbf{s})$, M computes a sequence of codes $\mathbf{v}_0 \to \mathbf{v}_1 \to \cdots \to \mathbf{v}_\ell$, and keeps track of a monomial $\mathbf{x}^{\mathbf{e}_0} \to \mathbf{x}^{\mathbf{e}_1} \to \cdots \to \mathbf{x}^{\mathbf{e}_\ell}$, where $\mathbf{e}_i \in \mathbb{N}^n$. The pair of indices (s_{i+1}, t_{i+1}) is used as the instruction to obtain $\mathbf{v}_{i+1}$ from $\mathbf{v}_i$, and $\mathbf{e}_{i+1}$ from $\mathbf{e}_i$.

To start, $\mathbf{v}_0 = \mathbf{v}$, and $\mathbf{e} = (0, \ldots, 0)$. Suppose at step i, $\mathbf{e}_i = (e_1, \ldots, e_n)$, and $\mathbf{v}_i = (v_1, \ldots, v_k, 0, \ldots, 0)$, $v_k \neq 0$. (k is the maximal nonzero index in $\mathbf{v}_i$.)

If $\mathbf{v}_i$ is anti-dominant, then $Y_{\mathbf{v}_i}$ equals to some Schur polynomial by 13 (1). Using Proposition 10 M can compute the evaluation of that Schur polynomial on $\mathbf{a}$ efficiently, and then multiply with the value $\prod_{i\in[n]} a_i^{e_i}$ as the output. Note that as will be seen below, the weight of $\mathbf{e}_i$ is $\ell \leq n \cdot |\mathbf{v}_0|^2$, so the bit length of $\prod_{i\in[n]} a_i^{e_i}$ is polynomial in the input size.

In the following $\mathbf{v}_i$ is not anti-dominant. M checks whether $t_{i+1} = k$. If not, M outputs 0.

In the following $t_{i+1} = k$. M then checks whether $s_{i+1} = t_{i+1}$.

If $s_{i+1} = t_{i+1}$, M goes to step $i+1$ by setting $\mathbf{v}_{i+1} = (v_1, \ldots, v_k - 1, 0, \ldots, 0)$, and $\mathbf{e}_{i+1} = (e_1, \ldots, e_{k-1}, e_k + 1, e_{k+1}, \ldots, e_n)$.

If $s_{i+1} < t_{i+1}$, then M tests whether s_{i+1} is an index in $\Psi_{\mathbf{v}_i}$, as follows. It first computes the permutation $\sigma := \langle (v_1, \ldots, v_{k-1}, v_k - 1, 0, \ldots, 0) \rangle \in S_N$, using the procedure described in Section 2. Note that $N = \max_{i\in[n]}\{v_i + i\} \leq |\mathbf{v}_i| + n \leq |\mathbf{v}_0| + n$. Then it tests whether s_{i+1} is in $\Psi_{\mathbf{v}_i}$, using Equation (4). If s_{i+1} is not in $\Psi_{\mathbf{v}_i}$, then M outputs 0. Otherwise, M goes to step $i+1$ by setting $\mathbf{v}_{i+1} = \mathfrak{c}(\sigma\tau_{s_{i+1},t_{i+1}})$, and $\mathbf{e}_{i+1} = \mathbf{e}_i$.

This finishes the description of M. Clearly M runs in time polynomial in the input size. M terminates within ℓ steps by Corollary 16. M always outputs a nonnegative integer as Schur polynomials are polynomials with positive coefficients.

Finally note that $Y_{\mathbf{v}}(\mathbf{a})$ is equal to $\sum_{\mathbf{s}} M(\mathbf{v}, \mathbf{a}, \mathbf{s})$, where $\mathbf{s}$ runs over all the sequences of pairs of indices as described at the beginning of the proof. By the discussion on #P in Section 2, this puts evaluating $Y_{\mathbf{v}}$ on $\mathbf{a}$ in #P. □

The Schubert basis is interpolation-friendly. Now we are in the position to prove that the Schubert basis is interpolation-friendly.

Proposition 17. $\{Y_{\mathbf{v}} \in \mathbb{Z}[x_1, \ldots, x_n] \mid \mathbf{v} \in \mathbb{N}^n\}$ *is K-bounded for $K(\mathbf{v}, n) = n^{2n\cdot(|\mathbf{v}|^2+|\mathbf{v}|)} \cdot \sqrt{|\mathbf{v}|!}$.*

Proof. The alternative proof of Corollary 3 for Schubert polynomials implies that $Y_{\mathbf{v}}$ can be written as a sum of at most $(n^2)^{n\cdot(|\mathbf{v}|^2+|\mathbf{v}|)}$ polynomials f, where f is of the form $\mathbf{x}^{\mathbf{e}} \cdot \mathrm{s}_\alpha$, $|\alpha| + |\mathbf{e}| = |\mathbf{v}|$. The coefficients in Schur polynomial of degree d are bounded by $\sqrt{d!}$ by Proposition 11. The claim then follows. □

Proposition 18. *$\{Y_{\mathbf{v}} \in \mathbb{Z}[x_1, \dots, x_n] \mid \mathbf{v} \in \mathbb{N}^n\}$ is L_A-compatible for $A = (r_{i,j})_{i,j\in[n]}$, $r_{i,j} = 0$ if $i < j$, and 1 otherwise. The leading monomial of $Y_{\mathbf{v}}$ w.r.t. L_A is $\mathbf{x}^{\mathbf{v}}$.*

The matrix A in Proposition 18 is the lower triangular matrix of 1's on the diagonal and below. Compare with that for Schur polynomials, described in Section 4.1.

Proof. This follows easily from Lemma 14: note that for any $\mathbf{c} = (c_1, \dots, c_n) \in (\mathbb{Z}^+)^n$, $\mathbf{u} \in Y_{\mathbf{v}}$, $\langle A\mathbf{c}, \mathbf{u}\rangle = c_1(u_1 + \cdots + u_n) + c_2(u_2 + \cdots + u_n) + \cdots + c_n u_n \le c_1(v_1 + \cdots + v_n) + c_2(v_2 + \cdots + v_n) + \cdots + c_n v_n = \langle A\mathbf{c}, \mathbf{v}\rangle$. As $c_i > 0$, the equality holds if and only if $\mathbf{u} = \mathbf{v}$. □

Now we conclude the article by proving the main Theorem 1.

Proof of Theorem 1. Note that $n^{2n\cdot(|\mathbf{v}|^2+|\mathbf{v}|)} \cdot \sqrt{|\mathbf{v}|!}$ is upper bounded by $2^{O(n\log n\cdot(|\mathbf{v}|^2+|\mathbf{v}|)+|\mathbf{v}|\log(|\mathbf{v}|))}$, and recall $|\mathbf{v}| = \deg(Y_{\mathbf{v}})$. Then combine Proposition 17, Proposition 18, and Theorem 9. □

Acknowledgement. Part of the work was done when Youming was visiting the Simons Institute for the program Algorithms and Complexity in Algebraic Geometry. We are grateful to Allen Knutson for his answer at `http://mathoverflow.net/q/186603`. We would like to thank the anonymous reviewers whose suggestions help to improve the writing of this paper greatly. Youming's research was supported by Australian Research Council DE150100720. Priyanka's research was supported by core grants for Centre for Quantum Technologies, NUS.

References

[ABS14] Sami Assaf, Nantel Bergeron, and Frank Sottile. A combinatorial proof that Schubert vs. Schur coefficients are nonnegative. *arXiv preprint arXiv:1405.2603*, 2014.

[AMS10] Vikraman Arvind, Partha Mukhopadhyay, and Srikanth Srinivasan. New results on non-commutative and commutative polynomial identity testing. *Computational Complexity*, 19(4):521–558, 2010.

[BB93] Nantel Bergeron and Sara Billey. RC-graphs and Schubert polynomials. *Experimental Mathematics*, 2(4):257–269, 1993.

[BC98] Nader H. Bshouty and Richard Cleve. Interpolating arithmetic read-once formulas in parallel. *SIAM J. Comput.*, 27(2):401–413, 1998.

[BF97] Alexander Barvinok and Sergey Fomin. Sparse interpolation of symmetric polynomials. *Advances in Applied Mathematics*, 18(3):271–285, 1997.

[BI13] Peter Bürgisser and Christian Ikenmeyer. Deciding positivity of Littlewood-Richardson coefficients. *SIAM J. Discrete Math.*, 27(4):1639–1681, 2013.

[BOT88] Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynominal interpolation (extended abstract). In Janos Simon, editor, *STOC*, pages 301–309. ACM, 1988.

[Bür00] Peter Bürgisser. *Completeness and reduction in algebraic complexity theory*, volume 7. Springer Science & Business Media, 2000.

[dCSW13] Cassio P de Campos, Georgios Stamoulis, and Dennis Weyland. A structured view on weighted counting with relations to quantum computation and applications. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 20, page 133, 2013.

[GKL11] Ankit Gupta, Neeraj Kayal, and Satyanarayana V. Lokam. Efficient reconstruction of random multilinear formulas. In *FOCS*, pages 778–787, 2011.

[GKL12] Ankit Gupta, Neeraj Kayal, and Satyanarayana V. Lokam. Reconstruction of depth-4 multilinear circuits with top fan-in 2. In *STOC*, pages 625–642, 2012.

[GKQ14] Ankit Gupta, Neeraj Kayal, and Youming Qiao. Random arithmetic formulas can be reconstructed efficiently. *Computational Complexity*, 23(2):207–303, 2014.

[Kay12] Neeraj Kayal. Affine projections of polynomials: extended abstract. In *STOC*, pages 643–662, 2012.

[KKL92] Adalbert Kerber, Axel Kohnert, and Alain Lascoux. Symbolic computation in combinatorics symmetrica, an object oriented computer-algebra system for the symmetric group. *Journal of Symbolic Computation*, 14(2):195 – 203, 1992.

[KM05] Allen Knutson and Ezra Miller. Gröbner geometry of schubert polynomials. *Annals of Mathematics*, 161(3):pp. 1245–1318, 2005.

[Kog01] Mikhail Kogan. RC-graphs and a generalized Littlewood-Richardson rule. *International Mathematics Research Notices*, 2001(15):765–782, 2001.

[Koi05] Pascal Koiran. Valiant's model and the cost of computing integers. *Computational Complexity*, 13(3-4):131–146, 2005.

[KS01] Adam Klivans and Daniel A. Spielman. Randomness efficient identity testing of multivariate polynomials. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *STOC*, pages 216–223. ACM, 2001.

[KS09] Zohar Shay Karnin and Amir Shpilka. Reconstruction of generalized depth-3 arithmetic circuits with bounded top fan-in. In *IEEE Conference on Computational Complexity*, pages 274–285, 2009.

[KY88] Erich Kaltofen and Lakshman Yagati. Improved sparse multivariate polynomial interpolation algorithms. In Patrizia M. Gianni, editor, *ISSAC*, volume 358 of *Lecture Notes in Computer Science*, pages 467–474. Springer, 1988.

[Las03] Alain Lascoux. *Symmetric functions and combinatorial operators on polynomials*, volume 99. American Mathematical Soc., 2003.

[Las08] Alain Lascoux. Schubert and Macdonald polynomials, a parallel. *Electronically available at `http://igm.univ-mlv.fr/~al/ARTICLES/Dummies.pdf`*, 2008.

[Las13] Alain Lascoux. Polynomials. *Electronically available at `http://igm.univ-mlv.fr/~al/ARTICLES/CoursYGKM.pdf`*, 2013.

[LS82] Alain Lascoux and Marcel-Paul Schützenberger. Polynômes de Schubert. *C. R. Acad. Sci. Paris Sér. I Math.*, 294(13):447–450, 1982.

[LS85] Alain Lascoux and Marcel-Paul Schützenberger. Schubert polynomials and the Littlewood-Richardson rule. *Letters in Mathematical Physics*, 10(2-3):111–124, 1985.

[LS03] Cristian Lenart and Frank Sottile. Skew schubert polynomials. *Proceedings of the American Mathematical Society*, 131(11):3319–3328, 2003.

[Mac91] Ian Grant Macdonald. *Notes on Schubert polynomials*, volume 6. Montréal: Dép. de mathématique et d'informatique, Université du Québec à Montréal, 1991.

[Man01] L. Manivel. *Symmetric Functions, Schubert Polynomials, and Degeneracy Loci.* Collection SMF.: Cours spécialisés. American Mathematical Society, 2001.

[MNS12] Ketan D Mulmuley, Hariharan Narayanan, and Milind Sohoni. Geometric complexity theory III: on deciding nonvanishing of Littlewood-Richardson coefficient. *Journal of Algebraic Combinatorics*, 36(1):103–110, 2012.

[MPP14] Karola Mészáros, Greta Panova, and Alexander Postnikov. Schur times Schubert via the Fomin-Kirillov algebra. *Electr. J. Comb.*, 21(1):P1.39, 2014.

[Mul11] Ketan Mulmuley. On P vs. NP and geometric complexity theory: Dedicated to Sri Ramakrishna. *J. ACM*, 58(2):5, 2011.

[Nar06] Hariharan Narayanan. On the complexity of computing Kostka numbers and Littlewood-Richardson coefficients. *Journal of Algebraic Combinatorics*, 24(3):347–354, 2006.

[Sap16] R. Saptharishi. A survey of lower bounds in arithmetic circuit complexity. `https://github.com/dasarpmar/lowerbounds-survey/releases`, 2016. Version 3.0.0.

[Shp09] Amir Shpilka. Interpolation of depth-3 arithmetic circuits with two multiplication gates. *SIAM J. Comput.*, 38(6):2130–2161, 2009.

[SV08] Amir Shpilka and Ilya Volkovich. Read-once polynomial identity testing. In *STOC*, pages 507–516, 2008.

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5:207–388, March 2010.

[Val79] Leslie G. Valiant. Completeness classes in algebra. In Michael J. Fischer, Richard A. DeMillo, Nancy A. Lynch, Walter A. Burkhard, and Alfred V. Aho, editors, *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 249–261. ACM, 1979.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In EdwardW. Ng, editor, *Symbolic and Algebraic Computation*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer Berlin Heidelberg, 1979.

[Zip90] Richard Zippel. Interpolating polynomials from their values. *J. Symb. Comput.*, 9(3):375–403, 1990.

A An alternative proof of skew Schubert polynomials in VNP.

By Equation Equation (1), skew Schubert polynomial can be written as:

$$Y_{\pi/\sigma}(\mathbf{x}) = Y_{\pi/\sigma}(x_1, x_2, \dots x_N)$$
$$:= \sum_C \mathbf{x}^{\mathbf{d}} / \mathbf{x}^{\mathbf{e}(C)} = \sum_C \prod_{i=1}^{N-1} x_i^{N-i-e_i} \tag{5}$$

Note that $e_N = 0$.

Let m be the length of an increasing chain. The first and second indices of each label in an increasing chain are encoded by $N \times m$ $0-1$ matrices g and b respectively. In these matrices the variables are listed along the row and the edges of a chain along the column. So for each column, there will be a 1 in the row which corresponds to the index that appears in that label. Thus in case of the g matrix it indicates which variable should be multiplied in the monomial.

All permutations are represented by $N \times N$ $0-1$ matrices, acting on length-N column vectors. W_0 and V, representing permutations σ and π respectively are given. Let $W_1, \dots W_m$ be the intermediate permutations in the increasing chain.

Now consider the following polynomials.

$$h_{1N} = \prod_{i=1}^{N} x_i^{(N-i)} \prod_{j=1}^{m} (\sum_{k=1}^{N} x_k^{-1} g_{kj}) \tag{6}$$

h_{1N} encodes the monomial for a given increasing chain, which depends on g.

In the following we shall gradually build up a series of polynomials, which basically characterize the property that g and b form an increasing chain.

$$h_{2N} = \prod_{t=1}^{m} [(\prod_{i,j,l,k} (1 - (W_t)_{ij}(W_t)_{lk})) \cdot (\prod_{i=1}^{N} \sum_{j=1}^{N} (W_t)_{ij})] \tag{7}$$

where the second product is over all $1 \leq i, j, l, k \leq N$ such that $i = l$ iff $j \neq k$. h_{2N} encodes valid permutation matrices. That is, it is non-zero iff $W_1, \dots W_m$ are valid permutation matrices, that is each row and column contains exactly one 1.

$$h_{3N} = [\prod_{i,j,k} (1 - g_{ik} g_{kj})] \cdot [\prod_{i,j,k} (1 - b_{ik} b_{kj})] \tag{8}$$

where both the products are over all $1 \leq k \leq m$ and $1 \leq i, j \leq N$ such that $i \neq j$. h_{3N} is non-zero iff each column of g and b has at most one 1.

$$h_{4N} = [\prod_{j=1}^{m} \sum_{k=1}^{N} g_{kj}] \cdot [\prod_{j=1}^{m} \sum_{k=1}^{N} b_{kj}] \tag{9}$$

h_{4N} is non-zero iff there is at least one 1 in each column of g and b.

$$h_{5N} = \prod_{i,j=1}^{N} [1 - ((W_m)_{ij} - V_{ij})] \cdot [1 + ((W_m)_{ij} - V_{ij})] \tag{10}$$

h_{5N} is non-zero iff $W_m = V$.

$$h_{6ijt} = \prod_{a,b=1}^{N} [1 - ((W_t)_{ab} - (W_{t-1}\tau_{ij})_{ab})].[1 + ((W_t)_{ab} - (W_{t-1}\tau_{ij})_{ab})] \cdot b_{jt} \tag{11}$$

h_{6ijt} is non-zero iff for a particular transposition (i, j) and for a pair of consecutive permutations W_{t-1} and W_t, (a) $W_t = W_{t-1}\tau_{ij}$ and (b) second index of the label is given according to definition of labeled Bruhat order.

$$h_{7t} = \sum_{i,j=1}^{N} \sum_{\substack{k,l=1 \\ k<l}}^{N} (W_{t-1})_{ik} \cdot (W_{t-1})_{jl} \cdot \prod_{\substack{i<a<j \\ k<b<l}} [1 - (W_{t-1})_{ab}] \cdot h_{6ijt} \tag{12}$$

h_{7t} is non-zero iff for any pair of consecutive permutations σ', π' being encoded by W_{t-1} and W_t respectively such that $\pi' = \tau_{ij}\sigma'$, we have $\sigma'(i) < \sigma'(j)$ $(i < j)$ and for every $i < k < j$, either $\sigma'(k) < \sigma'(i)$ or $\sigma'(k) > \sigma'(j)$. That is, $\sigma' < \pi'$ in the Bruhat order.

$$h_{8N} = \prod_{t=1}^{m} h_{7t} \sum_{i \leq s < j} g_{st} \tag{13}$$

where $1 \leq s < N$. h_{8N} is non-zero iff the sequence of permutations is correct, namely, maintaining the Bruhat order so that the labels are given accordingly.

$$h_{9N} = \prod_{t=1}^{m-1} \sum_{i=1}^{N} g_{it}[\sum_{j=i+1}^{N} g_{j,t+1} + g_{i,t+1}(\sum_{k=1}^{N} b_{kt} \cdot \sum_{l=k}^{N} b_{l,t+1})] \tag{14}$$

h_{9N} is non-zero iff each pair of labels in a chain respects the increasing lexicographic order.

We define the polynomial :

$$h_N(x_1, \ldots x_N, g, b, W_1, \ldots W_m) = h_{1N} \cdot h_{2N} \cdot h_{3N} \cdot h_{4N} \cdot h_{5N} \cdot h_{8N} \cdot h_{9N} \tag{15}$$

For each assignment to g, b, W_i, h_N is either 0, or a monomial corresponding to one correct increasing chain. It is clear that the size of a straight line program to evaluate h_N is polynomial in N. So h_N is in VP.

The skew Schubert polynomial, then can be given by

$$Y_{\pi/\sigma}(x_1, \dots x_N) = \sum_{g,b,W_1,\dots W_m} h_N(x_1, \dots x_N, g, b, W_1, \dots W_m) \tag{16}$$

g and b have mN elements and since m is $\mathcal{O}(N^2)$ so each has $\mathcal{O}(N^3)$ elements. Each W_i has N^2 entries. Thus the summation is over 0-1 strings of polynomial length.

This proves $Y_{\pi/\sigma}(\mathbf{x})$ is p-definable and hence is in VNP.